US Army Corps of Engineers
Construction Engineering Research Laboratories

USACERL Technical Report 96/32
January 1996

Dynamic Task Scheduling for the Knowledge Worker System

by
Gerald J. Brown, Milorad Sucur, and Wayne J. Schmidt

The Knowledge Worker Systems (KWS) is a performance support environment for workgroups. It is an automated tool that enables a workgroup to define the tasks, information resources, institutional knowledge and computer applications required to perform their business processes. Using this on-line model of the business process, KWS reminds workers when tasks are due, details steps for task execution, provides easy automated access to documents, and links to existing automation systems.

The U.S. Army Construction Engineering Research Laboratories has been conducting ongoing research into the problem of dynamic scheduling of processes and tasks for knowledge workers, with the ultimate goal of developing a comprehensive support environment for knowledge workers. KWS is a performance support environment designed to help knowledge workers organize and coordinate their work by providing an on-line model of the business process coupled with institutional knowledge and software agents. KWS tracks scheduled events, provides a list of completed events, and outlines the steps necessary to complete forthcoming tasks. This study examined the requirements for scheduling processes and tasks in a knowledge worker environment. Dynamic scheduling enhancements are identified for future versions of KWS.

USER EVALUATION OF REPORT

REFERENCE: USACERL Technical Report 96/32, *Dynamic Task Scheduling for the Knowledge Worker System*

Please take a few minutes to answer the questions below, tear out this sheet, and return it to USACERL. As user of this report, your customer comments will provide USACERL with information essential for improving future reports.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

___

___

___

2. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.)

___

___

3. Has the information in this report led to any quantitative savings as far as manhours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

___

___

4. What is your evaluation of this report in the following areas?

a. Presentation: ___

b. Completeness: ___

c. Easy to Understand: ___

d. Easy to Implement: ___

e. Adequate Reference Material: ___

f. Relates to Area of Interest: ___

g. Did the report meet your expectations? ___

h. Does the report raise unanswered questions? ___

i. General Comments. (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.)

5. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name:

Telephone Number:

Organization Address:

6. Please mail the completed form to:

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORIES
ATTN: CECER-TR-I
P.O. Box 9005
Champaign, IL 61826-9005

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1996	Final

4. TITLE AND SUBTITLE
Dynamic Task Scheduling for the Knowledge Worker System

5. FUNDING NUMBERS
4A162784
AT41
FF-AK5

6. AUTHOR(S)
Gerald J. Brown, Milorad Sucur, and Wayne J. Schmidt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratories (USACERL)
P.O. Box 9005
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER
TR 96/32

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, U.S. Army Corps of Engineers
ATTN: CEMP-MC
20 Massachusetts Avenue, NW.
Washington, DC 20314-1000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Knowledge Worker Systems (KWS) is a performance support environment for workgroups. It is an automated tool that enables a workgroup to define the tasks, information resources, institutional knowledge and computer applications required to perform their business processes. Using this on-line model of the business process, KWS reminds workers when tasks are due, details steps for task execution, provides easy automated access to documents, and links to existing automation systems.

The U.S. Army Construction Engineering Research Laboratories has been conducting ongoing research into the problem of dynamic scheduling of processes and tasks for knowledge workers, with the ultimate goal of developing a comprehensive support environment for knowledge workers. KWS is a performance support environment designed to help knowledge workers organize and coordinate their work by providing an on-line model of the business process coupled with institutional knowledge and software agents. KWS tracks scheduled events, provides a list of completed events, and outlines the steps necessary to complete forthcoming tasks. This study examined the requirements for scheduling processes and tasks in a knowledge worker environment. Dynamic scheduling enhancements are identified for future versions of KWS.

14. SUBJECT TERMS
Knowledge Worker System
computer programs
information management

15. NUMBER OF PAGES
38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Foreword

This study was conducted for Headquarters, U.S. Army Corps of Engineers (HQUSACE), Directorate of Military Construction, under Project 4A162784AT41, "Military Facilities Engineering Technology"; Work Unit FF-AK5, "Dynamic Task Scheduling." The technical monitor was John Sheehey, CEMP-MC.

The work was performed by the Business Processes Division (PL-B) of the Planning and Management Laboratory (PL), U.S. Army Construction Engineering Research Laboratories (USACERL). The USACERL principal investigator was Gerald J. Brown, and Wayne J. Schmidt was the project leader. Moonja P. Kim is Acting Chief, CECER-PL-B; L. Michael Golish is Acting Operations Chief, CECER-PL; and Dr. David M. Joncich is Chief, CECER-PL. The USACERL technical editor was Linda L. Wheatley, Technical Resources Center.

COL James T. Scott is Commander and Acting Director, and Dr. Michael J. O'Connor is Technical Director of USACERL.

Contents

SF 298 1

Foreword 2

Contents 3

1 Introduction 5
- Background 5
- Objectives 6
- Approach 6
- Mode of Technology Transfer 7

2 Dynamic Scheduling Issues 8
- Preplanning 8
- Task Scheduling 9
- KWS Scheduling Concepts 10
- Management of Resources 14

3 Current Dynamic Scheduling Implementation 16
- Dependent-Duration Activities 16
- Milestones 18
- Resource Leveling 19
- KWS Version 2.1 20
- KWS Version 2.5 21

4 Advanced Enhancements in Future Versions of KWS 23
- KWS Version 3.0 23
- KWS Version 4.0 24

5 Conclusions and Recommendations 25

References 26

Appendix A: KWS Data Structure 27

Appendix B: KWS Organizational Entities 34

Abbreviations and Acronyms 35

Distribution

1 Introduction

Background

Many Army personnel can be classified as *knowledge workers*—people who produce not tangible products, but some form of processed or enhanced information. Knowledge work is the area that offers the greatest opportunity to increase productivity within the U.S. workforce (Drucker 1974). While most Army knowledge workers depend on computer processing to complete their tasks efficiently, many do not have access to the overall process and its inherent schedule. No commercially available scheduling software was found that met the Knowledge Worker System (KWS) scheduling requirements: distribution of schedules, decentralized schedule logic, resource availabilities and consumption, or progress monitoring. Good schedule display and manipulation routines are available, but they are intimately bundled with data structures incompatible with KWS.

The U.S. Army Construction Engineering Research Laboratories (USACERL) has been conducting ongoing research into the problem of dynamic scheduling of processes and tasks for knowledge workers, with the ultimate goal of developing a comprehensive support environment for knowledge workers. KWS is an automated tool that enables a workgroup to define the tasks, information resources, institutional knowledge, and computer applications required to perform their business processes. KWS tracks scheduled events, provides a list of completed events, and outlines the steps necessary to complete forthcoming tasks.

Issues in dynamic scheduling for KWS were discussed for the first time in a report by Thomas and Schmidt (1992). A KWS group planning meeting was held 5–6 October 1992 at USACERL. In attendance were George Olive and Mike Jones, researchers at the Construction Research Center, Georgia Institute of Technology (Georgia Tech). The group discussed the following scheduling and resource issues:

- security in accessing and modifying task information
- notification of schedule assignments, changes, and status of dependant tasks
- determination of priority level, due date, and dependencies
- task and project status and summary reports

- scheduling algorithm that reflects dependencies, hierarchical structure, and cycles
- bidirectional (representation and output) graphical representation of schedule, dependencies, and status
- tracking global and individual manpower availability
- task duration estimation
- generation of job description and performance reporting
- simulations and what-if analysis on scheduling of resources.

A meeting was next held at Georgia Tech 19 November 1992 to discuss scheduling issues and the current scheduling algorithm. After a discussion of a bottom-up definition of tasks versus a top-down decomposition of processes, the group concluded that top-down decomposition was preferable from a manager's perspective, so Coe Truman Technology (CTT) was contracted to develop a top-down model using Integrated Definition Language (IDEF) (Paragon 1993) to see how the model output could serve as input to KWS. IDEF is primarily intended for process analysis.

On 26 April 1993 another meeting was held at Georgia Tech to discuss scheduling enhancements to KWS. USACERL contracted with Dr. Donovan Young of the School of Industrial and Systems Engineering at Georgia Tech, to research scheduling enhancements to KWS. Dr. Young produced *Scheduling Enhancements to Knowledge Worker System* (December 1993), which also presented scheduling data structures. On 18 November 1993, Dr. Young presented the results of his scheduling research to a working group that included the technical monitor for KWS, John Sheehey. These meetings, investigations by the principal investigator, and input from KWS team members formed the basis for identifying dynamic task scheduling requirements for KWS.

Objectives

The objective of this work was to develop the dynamic scheduling program requirements that will be incorporated in KWS.

Approach

A dynamic scheduling system for KWS was developed in three phases. Phase 1 consisted of brainstorming to develop initial knowledge worker scheduling requirements and incorporating them in a prototype version of KWS. Phase 2 was primarily Dr. Young's research as well as evaluating off-the-shelf scheduling systems. Phase 3

consisted of designing the scheduling system to include data structures and screens, which will be incorporated in a final version of KWS to be implemented by Georgia Tech.

Mode of Technology Transfer

Dynamic scheduling is an integral component of the complete Knowledge Worker System. The scheduling features of KWS are being tested at several pilot sites:

- U.S. Army FORSCOM/7th Transportation Group
- Fort Eustis Directorate of Public Works
- Defense Logistics Agency
- Corps of Engineers Military Programs Directorate.

2 Dynamic Scheduling Issues

The rapid growth in the availability and power of microcomputers has made it possible for knowledge workers to monitor and control the progress of many interrelated tasks (East and Kirby 1990). The bar (or Gantt) chart provides a visual means of measuring performance against goals. Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) were developed in the 1950s. Today, more than 200 PC-based scheduling packages are available (O'Brien 1971). All scheduling systems have several common features such as early and late start times, finish times, and slack times for activities. KWS extends the concept of traditional project management, which focuses on planning individual isolated projects, to an enterprise-wide consideration of multiple projects, shared resources, and mission goals. KWS requires dynamic scheduling as part of the core program. Appendix A shows the KWS data structure.

Preplanning

Before entering schedule data into KWS, it is wise to do some preplanning. A "top-down" approach to project or process planning is encouraged. Rather than brainstorming or listing tasks from the lowest level, top-down is a structured approach to initializing a schedule. Some IDEF models may be available as a starting point. IDEF also provides a Work Breakdown Structure (WBS), which helps in preplanning.

KWS keeps information about an organization's projects or tasks in a hierarchical format. Basically, a project is divided into several processes, which are then subdivided into tasks. Many government offices may have a numbering system for processes and tasks. Major processes will be added to KWS most often by the Business Process Reengineering (BPR) team in collaboration with affected knowledge workers.

The tasks within one process may be performed by many different knowledge workers. KWS tracks jobs and their associated due dates by using a master scheduler program. When a process is assigned, the scheduler adds that process to the KWS database. Appendix B has detailed definitions of all organizational entities in Knowledge Worker System. Figure 1 is an example of a process and related tasks.

Process: Manage Division Resources

1. Manage Equipment Inventory
   - Log Equipment
   - Update Inventory List
   - Complete Property Loan Receipts
   - Scan Equipment
   - Resolve Inventory Discrepancies

2. Schedule Meeting Rooms
   - Check Meeting Room Availability
   - Reserve Meeting Room
   - Find Alternate Meeting Room
   - Notify Requester

3. Monitor Division Supplies
   - Log Request for Supplies
   - Request Supplies
   - Notify Requester
   - Distribute Supplies

4. Produce Signs
   - Format Sign Information
   - Print Sign
   - Assemble Sign
   - Distribute Sign

Figure 1. Example process and tasks.

Task Scheduling

Once the process and tasks are defined, task scheduling requires planning for resources, durations, and sequencing.

Resources

The knowledge worker(s) that perform the task are resources. Availability is a planning consideration. If a knowledge worker is assigned more than one task, the KWS scheduler allows specification of the percent of total effort spent on the task each day. A task priority can be used to specify which tasks should be worked on first when two or more tasks are assigned to a knowledge worker.

Task Durations

Estimates of task duration are required to determine the overall process time. Each task takes a certain number of days or hours, and tasks must be scheduled according to the resources available. Some tasks such as decision points or event occurrences have no time associated with them. These points are referred to as milestones. Multiple duration estimates (e.g., most likely, pessimistic, and optimistic) required for PERT analysis are cumbersome to use and not as widely used as CPM single-duration estimates.

Sequencing

Sequential tasks may be arranged by (1) assigning priorities and leveling the process or (2) entering dependency links, which is known as schedule logic. Some tasks can be performed in parallel, while others must start or finish in a certain order. The scheduler allows the user to establish these links. When planning a process, tasks and their logical task sequence should be considered. Task scheduling includes prioritizing and linking dependent tasks, as well as basic information such as duration and due dates. Resource scheduling includes assigning employees to tasks and prioritizing when several tasks vie for an employee's time. Sometimes it is necessary to reassign tasks in order to balance employee workloads.

KWS Scheduling Concepts

Figure 2 shows the cascading work hierarchy. Each process drives an organizational responsibility and has a due date. Processes break down into tasks for which durations can be specified. Tasks can be further decomposed into other tasks. All this information can be considered the "what" and "when" drivers for knowledge workers. The priority level for tasks needs to be determined and presented to the user. This priority level should reflect not only the priority assigned the task by supervisors, but also the imminence of the deadline. As an additional visual guide, tasks will be color-coded by priority.

A shared, distributed schedule is an essential component of KWS. This master schedule tracks the tasks for which each knowledge worker is responsible. Task information will be maintained in a database. The system must maintain estimates of task durations, track their actual durations, and retain this information for future reference. This information can later guide estimates of task durations. KWS will track several categories of tasks: one-time-only tasks,

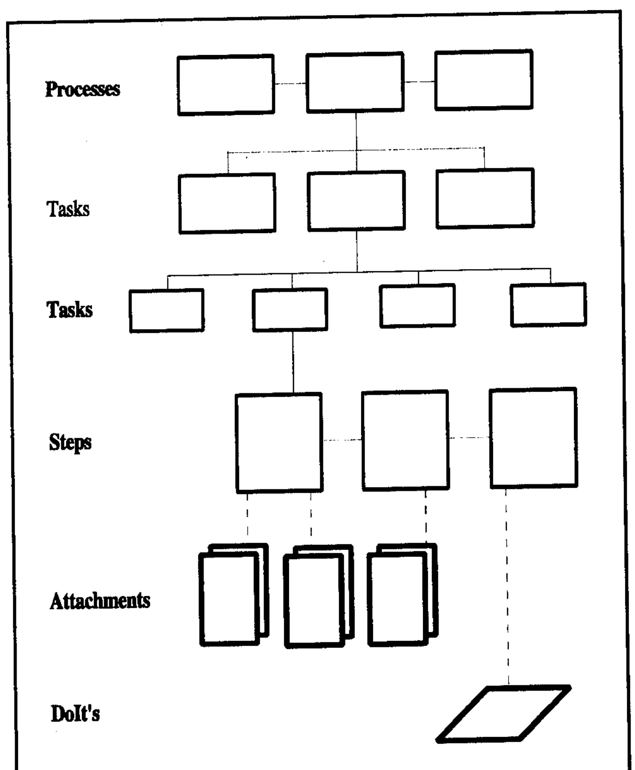

Figure 2. KWS process hierarchy.

periodically repeated tasks, and sporadically repeated tasks. As an added benefit, KWS provides a repository of processes and tasks for reuse as appropriate.

One-Time-Only Tasks

These tasks are nonrepetitive tasks that often occur throughout the year on an *ad hoc* basis. Examples might be to staff a suspense, prepare an after-action report, develop a new concept, or respond to a data request from headquarters.

Periodically Repeated Tasks

To create periodically repeated tasks (or cyclic tasks in KWS terminology), the user follows the same procedure as for nonrepeatable tasks but marks the "Cyclic" field in the task definition screen (shown in Figure 3). Then, the cycle period (i.e., weekly, monthly, quarterly, or yearly) and the time of the last cycle should be chosen, all in the same window. When a periodically repeated task is created, a template used for the creation of additional cycles of the task is also created. Changes to the template will

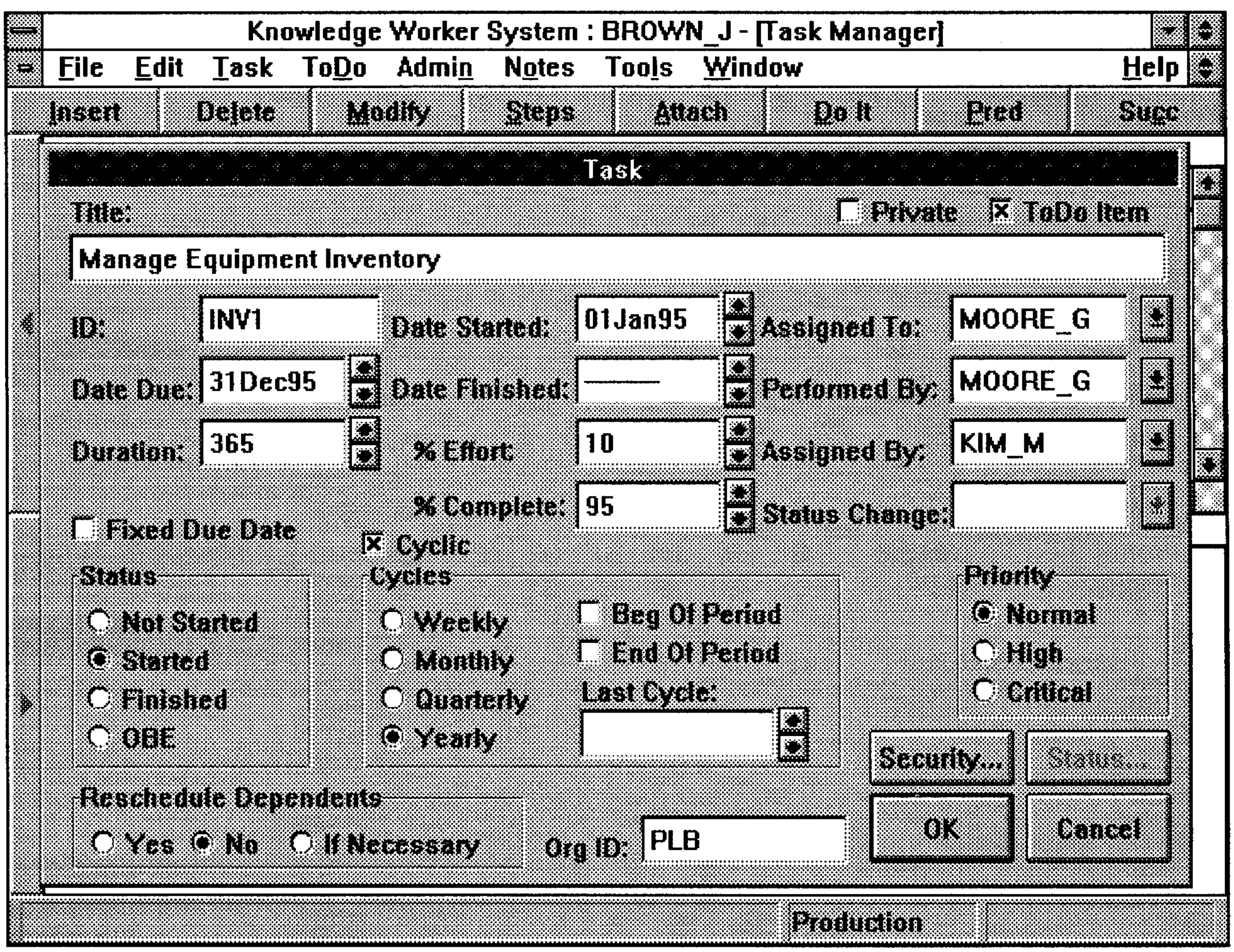

Figure 3. Task input form.

be reflected in any new iterations of the task but will not affect a cycle that existed before that change. A noncyclic task can be changed to a cyclic task, but a change in the opposite direction is not allowed, because cyclic tasks require more definition and cannot be simplified.

Sporadically Repeated Tasks

These tasks can be created and expanded using templates created by users from some noncyclic tasks or from automatically created templates of some periodically repeated tasks. KWS will allow users to modify a task's procedure list to reflect unique circumstances—procedures can go back to being done the usual way the next time without extra effort. Of course, users also will be able to modify the usual procedure (e.g., if they find a more efficient method).

Each knowledge worker will be able to view the progress of others who must complete work before his/her task can begin. Conversely, each worker also will be able to determine when other workers in the group need his/her piece of the process to be done. The system will provide a graphical representation of schedule, dependencies, and status, which will serve as a means of presentation as well as input. Figure 4 shows an example of the current graphical scheduling interface. Task structure can also be entered as text in the event manager.

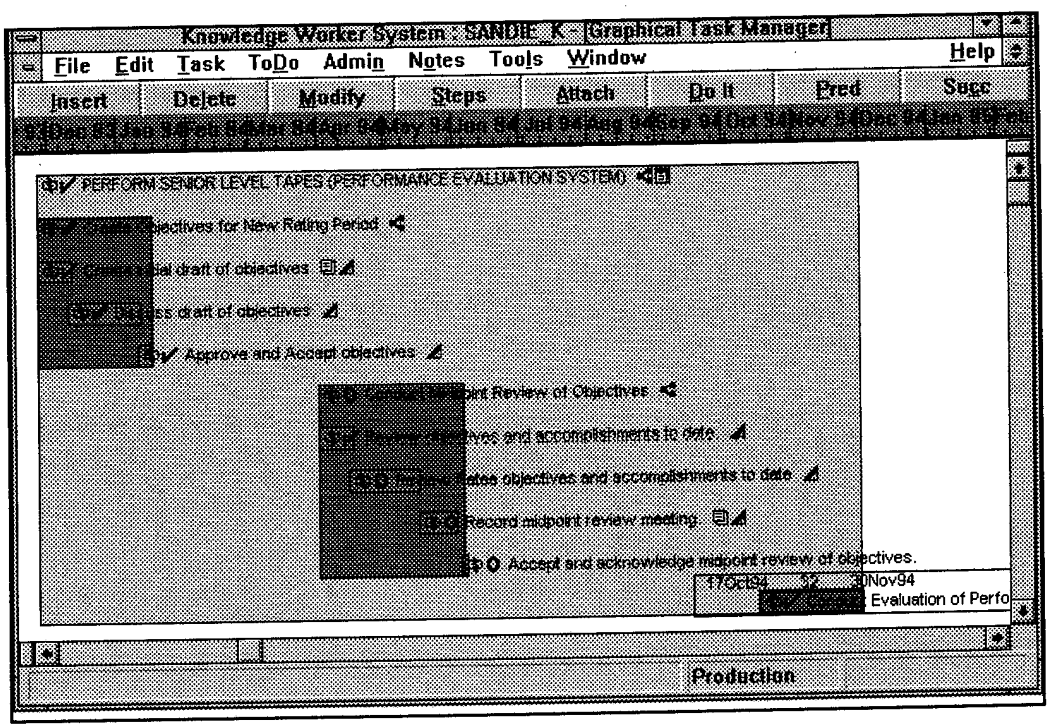

Figure 4. Graphical task manager.

A knowledge worker's daily activities are part of a larger process. Generally, higher authorities determine the overall process and calendar; thus, the knowledge worker has little control over timing and due dates. The knowledge worker initiates actions, analyzes information, and produces products according to the preset schedule.

Though the master schedule is preset, it is not permanently fixed. As the task progresses, it must be periodically replanned, rescheduled, and recoordinated. KWS will accommodate changes in the work schedule by calculating their effects on related activities, adjusting due dates, and providing information about the system-wide effects of the change.

Task Assignments

After a task (or process) has been entered into the database, it may be necessary to reassign the task to another worker. KWS contains three assignment fields: "Assigned To," "Performed By," and "Assigned By." The "Assigned To" and "Performed By" fields can be changed at any time after that task has been added to the database. When these fields are changed, KWS will automatically reassign the task to the new knowledge worker's ToDo List. It is possible to reassign tasks even if the original task owner has marked it "Started" in the status field. The "Assigned By" field contains the Group or User ID of the knowledge worker who assigned the work.

Task Security

The security button on the Task Input Form brings up the Task Security dialog box. This dialog box is used to define the security level for this task for all knowledge workers in the organization.

Delete — The knowledge worker or group has permission to delete the task from the list.

Update — The knowledge worker or group has a set of update privileges.

Assignment — The user may change the "Assigned To" and "Performed By" fields of the task.

Due Date/ Duration — The user may modify the due date and duration of the task.

Status — The user is allowed to modify the task status.

All other attributes The user is allowed to modify all the other attributes of the task.

By default, the task creator has full permission and the public has permission only to modify status. The task creator can select any combination of permissions.

Management of Resources

KWS will use a task database to store schedules and other information. This approach makes a central database available to other programs within the Corps of Engineers Application Program (CEAP) environment for management reporting (e.g., tracking by functional area and division). Output from KWS can be in the form of reports or files for use in other Corps of Engineers applications.

The schedule can be viewed in a variety of ways. It will be presented to knowledge workers as a graphical or textual list of tasks for which they are responsible. They can also "take a step back" to better understand their group as a whole. Supervisors can use this information to assess the effect of personnel shifts, to implement new projects and manage existing ones, and to plan and monitor resource changes. Project status and summary reports will also be available.

The system will be able to display available manpower as a histogram (Figure 5), which would allow supervisors to see the workload of knowledge workers at a specified time. They can use this information to help balance the workload. As dates within the schedule change, the affected knowledge workers will be notified electronically. All documents, files, and executables associated with the tasks affected by the schedule change will also be shifted. This linkage will ensure that all information pertinent to task performance is readily accessible to the knowledge worker to whom the task is transferred.

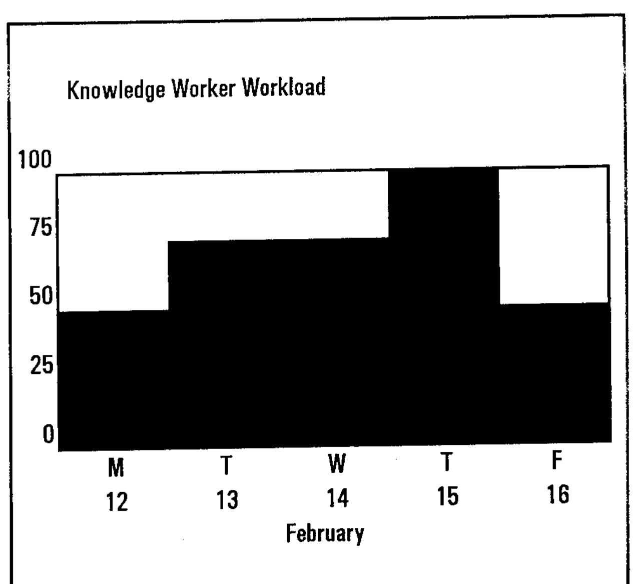

Figure 6. Resource histogram.

Similarly, when a supervisor reassigns a task to another member of the workgroup, all associated information will be transferred with it. In this way, the newly assigned knowledge worker receives the institutional knowledge of

previous performers of the task, including their productivity tools and relevant documentation. The system will be able to generate job descriptions and performance requirements from historical ToDo list information. This historical information will also provide a basis for estimating an individual knowledge worker's skill at performing different tasks.

3 Current Dynamic Scheduling Implementation

Dr. Donovan Young of Georgia Tech developed a conceptual design for enhancements to the scheduling capabilities of KWS. In a final report presented to the USACERL KWS team and the KWS technical monitor on 18 November 1993, Dr. Young envisioned a dynamic task scheduling system for a distributed workgroup of knowledge workers that would provide a personalized view of their tasks within the enterprise mission. This view would include not only ToDo lists, but also a set of more advanced functions associated with resource-constrained scheduling. This capability would allow convenient and accurate coordination of widely distributed activities. Dr. Young's analysis had three primary recommendations: dependent-duration activities (DDAs), milestones, and resource leveling. Other enhancements were identified by KWS team members and feedback from pilot sites.

Dependent-Duration Activities

A task is a lowest-level scheduling primitive that has duration, ownership, attached resource(s), and priority. Each knowledge worker can perform one or more tasks. When creating a new task, a knowledge worker defines the task parameters on the task definition screen (Figure 2).

A task is a single operation that occurs over the course of the project. A DDA is a logical grouping of tasks. DDAs represent groups of activities that can be in as many levels as desired.

Each DDA has scheduled start and finish times determined by the mininima and maxima of the start and finish times of its members. No task can have more than one parent in the DDA hierarchy. DDAs have no schedule-relevant problem data except their lists of members; precedences for all members of a DDA can be assigned by specifying precedences for the DDA. Figure 6 shows a DDA and its member tasks. To keep the screen from becoming cluttered as a process is entered, lower levels of a DDA can be hidden or shown as needed in the KWS Graphical Task Manager (GTM).

DDA Manage Equipment Inventory
Task Log Equipment
  Task Update Inventory List
    Task Complete Property Loan Receipts
      Task Scan Equipment
        Task Resolve Equipment Discrepancies

Figure 6. An example DDA and its member tasks.

In the current version of KWS, tasks can be subdivided in the desired number of task levels, which roughly corresponds to the definition of DDA. Figure 7 shows a graphical representation of a higher level task and its subtask hierarchy in the GTM. If the task in the GTM is shown with a "+" sign, it has associated subtasks that are not currently shown in the GTM. When the user expands the task in GTM, the "-" shown denotes that all subtasks of the task are shown in the GTM window.

DDA is a grouping of the sequence of tasks that contains the institutional knowledge about how some tasks are to be performed. If there are tasks (on any level of organizational hierarchy) that can be used potentially by some other knowledge workers, the knowledge worker that initially created the task can make it available

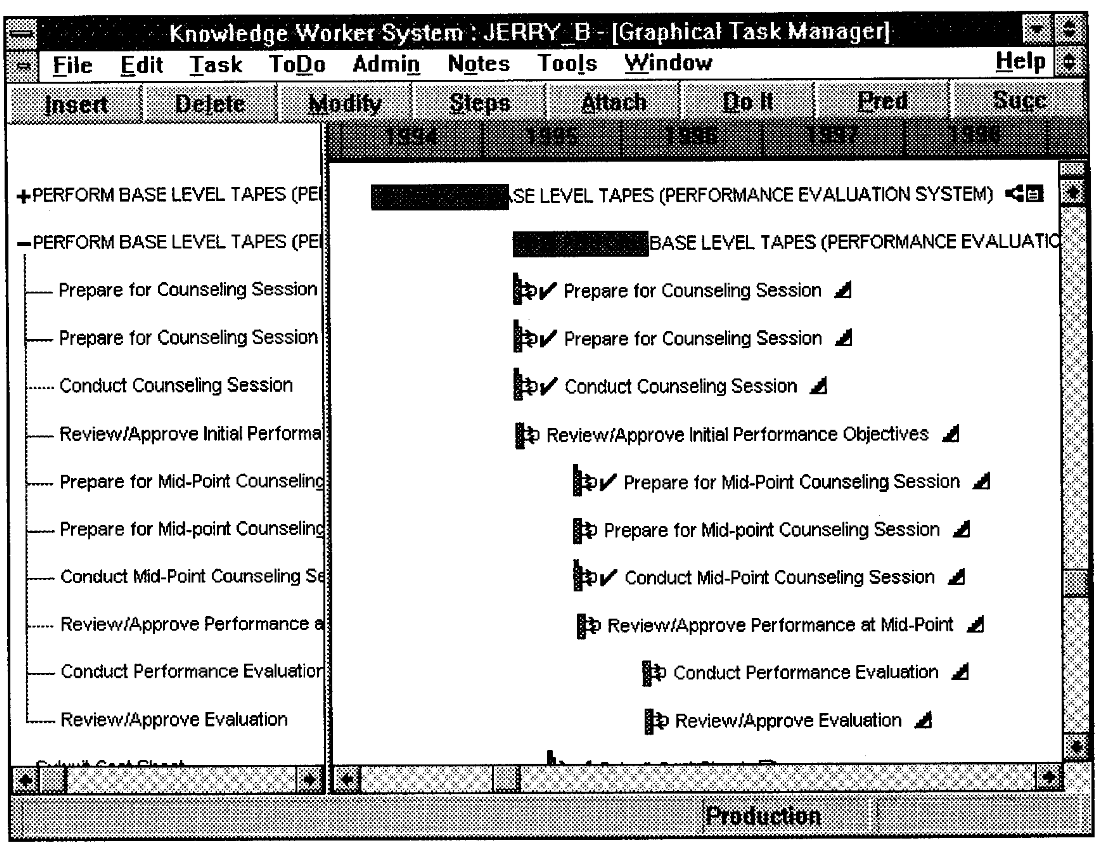

Figure 7. Subtask display.

to all others through the use of Task Palette (shown in Figure 8) where task templates are stored. In that way, the institutional knowledge about some business processes is retained and reused, which promotes Business Process Improvement (BPI) and Total Quality Management (TQM) in the organization through continued use of KWS. A task template not only retains institutional knowledge but also establishes a process discipline. A "Process Owner" is designated, and only that owner may update, change, or improve the process. However, by reusing the process templates and crafting them to specific needs of newly created tasks and processes, continual process improvement is supported. In this way the old task templates serve as "seeds" for the creation of new tasks by efficiently using previous process experience and knowledge.

Milestones

Milestones are tasks with zero duration. They serve as markers to indicate that an event is taking place, such as marking process completion or a management review. Milestones are distinct from tasks with a special graphical symbol. Milestones are shown as up-side-down arrows. If an owner delays the scheduled time of one or more milestones, the schedules for other workers will show the new scheduled time.

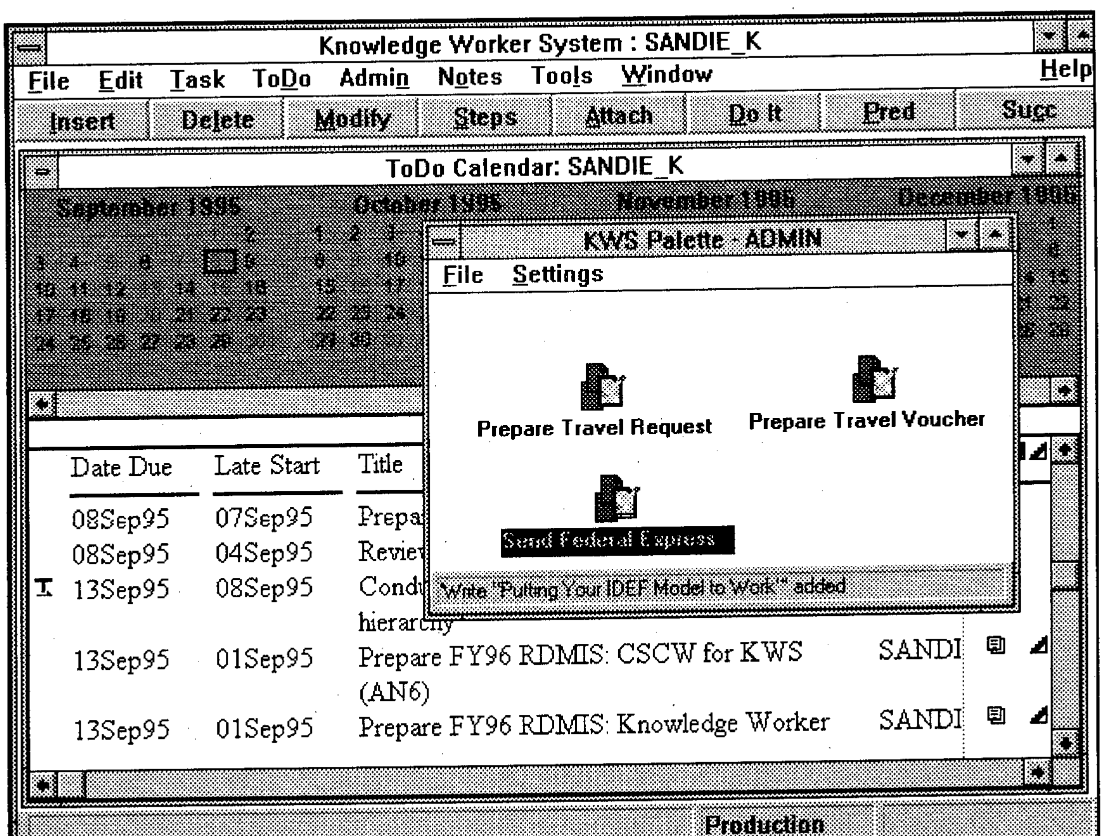

Figure 8. Task Palette.

If a worker has scheduling authority for a task that is a successor of a newly delayed milestone, this worker will control the way the predecessor milestone's delay affects his/her schedule. A task in a precedence string affected by the delay will automatically be delayed if its start time has been left free or will show negative slack if its start time has been fixed.

Milestones are supported in the current implementation of KWS. To view the organizational milestones, the user should choose "Task" from the main menu and "Milestones" from the choices available under it. The window showing the organizational milestones will appear (Figure 9). To view the successor and predecessor tasks of a milestone, the user selects the milestone in an open milestone window and chooses "Succ" or "Pred" on the button bar. Specifying and managing successor and predecessor relationships for a milestone is also facilitated through a graphical user interface in GTM. KWS version 2.5 will fully support the successor and predecessor logical relationship for milestones.

Resource Leveling

KWS Version 2.0 allows assignment of resources to a task, but does not keep track of resource availability or conflicts. Because the duration of a task depends on the rate at which effort is applied, effective scheduling cannot be done without representation of resources, except in special cases (e.g., every task is to be done by a single worker

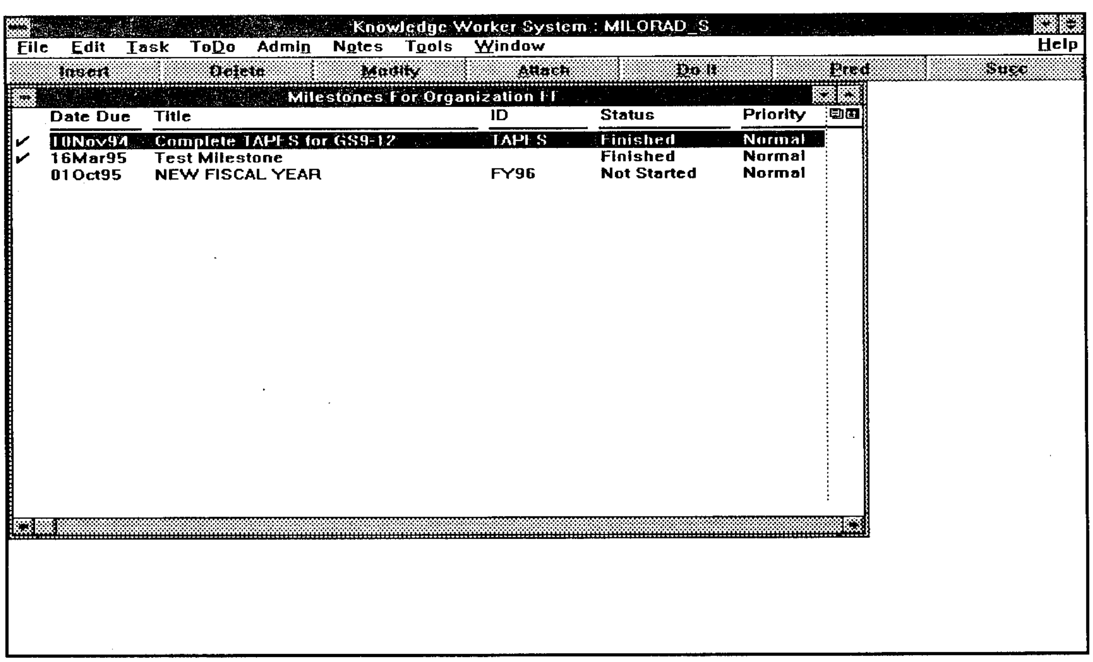

Figure 9. Milestones.

at a fulltime rate, which is not how most professional work is done). The enhanced KWS will include representation of the availabilities and consumptions of resources.

Resource data enters the task database both by explicit declaration and by default. The resources that exist by default are the knowledge workers who are members of the workgroup. Each of these has by default an identifier and a standard availability profile. Examples of resources include the worker; another worker in the workgroup; or team members who are considered more or less interchangeable.

Each task that consumes a given resource is considered to consume a given amount over the task's duration, and the resource amount is measured in resource units (e.g., man-days). Further, the consumption rate of the resource by the task is considered to be constant over the task's duration, and the consumption rate is measured in rate units (e.g., man-days per day). For example, if a 30-day task consumes 15 man-days of knowledge worker time, it is considered to consume 0.5 man-days each day. Each resource has an availability, and that availability is the amount of consumption that should not be exceeded during the task.

A resource conflict graph would show resource conflicts for all resources. The resource data table lists the availabilities, consumptions, and conflicts for all tasks and resources. It can be used to guide scheduling actions (e.g., to remind the scheduler which activities consume a given resource and may be candidates for reassignment). A knowledge worker's daily hours may be distributed among several different tasks.

KWS Version 2.1

KWS Version 2.1 has been tested and is currently being installed at pilot test sites. Scheduling enhancements which were identified and programmed are as follows:

1. *Develop capability to define process with both cyclic and noncyclic tasks.* Currently no capability exists to define a process that includes both cyclic and non-cyclic tasks. The capability to place cyclic under noncyclic tasks needs to be developed. It should be possible to place cyclic tasks with varying frequency of occurrence under noncyclic tasks. The number of cycles in child tasks has to be based on the duration of a parent task and the duration of one cycle of a child task. If the duration of the parent cyclic task is changed by the user, the number of cycles needs to be changed accordingly. Allow the user to override the computed number of cycles in the child task with the constraint that the duration of the parent task must not be exceeded.

2. *Develop capability to insert a task for a single nonstandard cycle of a cyclic task.* An example of a cyclic task is "Attend Weekly Reviews." A child task for a single nonstandard cycle that may be inserted within that parent task is "Prepare a Mid-Year Review."

3. *Enhance parent/child relationship to include the proper inheritance of status.* This capability pertains to propagation of change of status from child to parent while forbidding the propagation of change in the direction from parent to child. For example, if all child tasks are marked "Complete" or "Overcome by Events (OBE)," the parent task displays "Complete" or "OBE" status. When the status of a child task is later toggled to "Started" or "Not Started," the parent must reflect the change. The status of child tasks must not change if the user tries to toggle the status of the parent task to "Started" or "Not Started." In that case, a warning needs to display explaining that the only way to change the parent task status is by modifying the status of a child task.

KWS Version 2.5

KWS Version 2.5 is currently under development and will undergo testing in March 1996. Scheduling enhancements which have been identified and are being programmed are as follows:

1. *Enhance the scrolling feature in the graphical task manager.* The scrolling feature needs to be modified so that when a task duration is changed in GTM by dragging the task bar, the whole task window will scroll together the position of the mouse arrow.

2. *Enhance the task duration input interface.* The task duration input interface needs to be changed so the user can enter task duration as the combination of days and hours. The modification of the input field needs to be done so *Days* and *Hours* can be entered separately.

3. *Develop the capability to enforce predecessor/successor relationships.* Once the predecessor/successor relationship is defined, the start date of the successor task is to be set to the due date of the predecessor task. The scheduling system must not allow a user to mark a successor "Finished" before its predecessor is complete. Likewise, the system must not allow the successor tasks to be marked "Started" before predecessor tasks are finished.

4. *Improve the functionality of Task Identifications (IDs).* Current KWS users are often not using Task IDs for identifying tasks, subtasks, and duplicate tasks generated using the Task Palette. The following changes are needed:

a. When creating a process, force the user to enter a Task ID

b. When creating a child task, use the parent Task ID as the default. The user may change the default value of the child task, but some entry is required.

c. Modify Task Palette operations so that when dropping a task into KWS, the user is prompted with the attribute title and variable value. Do not allow the user to put blank data in the title or ID field. If the task is dropped into KWS under a parent task, the default value of $ID becomes the Task ID of the parent task. Prompt the user to link an identifier to the title and all subtask titles in order to differentiate task titles.

5. *Improve the graphical view of scheduling data.* Information about the amount of positive or negative slack the task has is not displayed at present. The only information about lateness or earliness of tasks is conveyed through displaying the tasks that are late in red and categorization of tasks that are not started as "In Danger of Becoming Late."

6. *Implementation of milestones.* Develop detailed specification for the implementation of milestones. In particular, develop the capability to allow the user to export a portion of the KWS schedule, manipulate it in the local database, and upload to the master schedule. Include the addition of new data fields to display slippages and notification of milestone violations.

4 Advanced Enhancements in Future Versions of KWS

This chapter outlines the upgrades for dynamic scheduling enhancements to KWS. Scopes of Work (SOWs) and contracts have been developed for Version 3.0, which will be developed in fiscal year 1996. KWS Version 4.0 requires additional research before a scope of work can be developed. The remainder of this chapter discusses KWS versions and dynamic scheduling features.

KWS Version 3.0

1. *Develop the capability of sharing tasks across organizations.* Develop the capability to allow knowledge workers to share tasks across multiple organizations and to track the tasks once assigned to another organization. For version 3.0, this capability will apply to organizations that share a common database server. The development of this capability includes:

 a. Providing the command under the <ADMIN> menu enabling a two-level browsing of organizations and knowledge workers in those organizations.
 b. Allowing assignment of knowledge workers from different organizations to a single workgroup.
 c. Allowing any part of a process to be assigned to knowledge workers in different organizations. No limit will exist for the number of different organizations that a task can be assigned to within a process.

2. *Provide status icon for tasks waiting predecessor completion.* A capital <W> status icon on the left side of the knowledge worker ToDo window will denote that a successor task cannot be completed because it is waiting for the predecessor task to be completed.

3. *Enhance management of public/private task attribute.* A task inserted from the ToDo calendar will be automatically considered private while tasks inserted from the task manager or GTM will be considered public. Any private task can be changed to public by dragging the task to a task manager/GTM or by dragging to the Task Palette. The radio button <PRIVATE> in the task dialog box in version 2.X will be removed.

4. *Enhance resource scheduling by expanding capabilities of specifying resource availability information.* It is assumed that there are 8 hours in a work day and that Saturdays and Sundays are not working days. To allow more flexibility in resource scheduling, knowledge workers need to be able to change their default working hours. To provide for this flexibility, an option under the <ADMIN> menu should be added called <Change Working Time>. In this option, the ToDo calendar will be displayed, but the Task and Appointment calendar will be omitted. At the bottom of the window, the scrolling options and a button will allow knowledge workers to specify working, nonworking, and default values and times. Nonworking days can be displayed in reverse video on the ToDo calendar. The option should be given to the knowledge worker to choose whether he/she will perform the task that was just inserted on a weekend or holiday or postpone it to a normal work day. If a process is moved or dragged from the Task Palette involving a large number of tasks, the tasks should be automatically rescheduled to work days only rather than prompting for every non-working day conflict.

5. *Color coding and icons will be added to clarify schedule status.* Critical priority tasks shall appear in red; high priority tasks, blue; normal priority tasks, black. Stoplight-like icons shall be displayed in the left-most field adjacent to the task title and shall be red for late tasks; yellow for tasks in danger of being late; and green for tasks available to be started. For tasks waiting on a precedent task and therefore marked with a "W," the "W" will be overlayed in red in cases where the task is late.

KWS Version 4.0

KWS Version 3.0 applies to organizations that share a common database server. For KWS Version 4.0, further research is required to determine scheduling requirements in a multiserver environment. Milestones are the mechanism for transmitting information between servers, but data transferral and display features need to be clarified. In addition, some consideration should be given to downloading an individual's KWS schedule to a portable computer for field work, and uploading the KWS schedule when the worker returns.

Version 3.0 specifies the implementation of individual resource scheduling and utilization reports. These features may need to be refined and extended in cases when intelligent resource leveling is required due to limited resources. Finally, KWS scheduling should be investigated to determine how historical task data retained in KWS could best be used. In KWS Version 4.0, historical data may be used to estimate duration of similar tasks or for knowledge worker job descriptions and evaluation reports (percent tasks completed on time, etc).

5 Conclusions and Recommendations

This study concludes that it is feasible to develop a dynamic scheduling environment for KWS. Early versions of KWS have been field tested, and the enhancements noted should add significant scheduling features. The KWS environment is much more dynamic than the scheduling environment for construction management, which may take several months to develop. The KWS performance support environment allows shared task support for the coordination of knowledge workers in dynamic, increasingly complex environments. Knowledge workers work under a highly event-oriented and date-driven schedule and may work as groups or even at different sites, requiring a distributed database. Besides maintaining the schedule, the addition of attachments and DoIts to tasks is a substantial benefit to knowledge workers, providing procedurial information and automating repetitive processes. Dynamic scheduling enhancements will be incorporated in future versions of KWS.

References

Drucker, Peter F., *Management* (Harper & Row, 1974).

East, E. William, and Jeffrey G. Kirby, *A Guide to Computerized Project Scheduling* (Van Nostrand Reinhold, 1990).

O'Brien, James J., *CPM in Construction Management* (McGraw-Hill, 1971).

Paragon Systems, Inc., *IDEF0 Reference Manual* (1993).

Thomas, Beverly E., and Wayne J. Schmidt, *Building a Knowledge Base for the Knowledge Worker System*, Interim Report FF-92/02/ADA258544 (USACERL, August 1992).

Young, Donovan, *Scheduling Enhancements to the Knowledge Worker System* (Georgia Institute of Technology, December 1993).

Appendix A: KWS Data Structure

```
CREATE TABLE KW (
    KW_ID            VARCHAR(20) NOT NULL,
    KW_FIRSTNAME     VARCHAR(40),
    KW_MIDDLENAME    VARCHAR(40),
    KW_LASTNAME      VARCHAR(40),
    KW_SUPERVISOR    VARCHAR(20),
    KW_DEPT          VARCHAR(40),
    KW_PHONE         VARCHAR(20),
    KW_FAXPHONE      VARCHAR(20),
    KW_ADDRESS       VARCHAR(240),
    KW_ADMIN         INTEGER,
    KW_MSGFLAG       INTEGER,
    KW_GROUP         VARCHAR(10),
    KW_ORG           VARCHAR(20),
    KW_MSGCHECK      DATE
    );
CREATE UNIQUE INDEX KW_ID ON KW(KW_ID);

CREATE TABLE ID_COUNTERS (
    TASK_COUNTER  INTEGER,
    STEP_COUNTER     INTEGER,
    ATTACH_COUNTER   INTEGER,
    DOIT_COUNTER     INTEGER,
    MESSAGE_COUNTER  INTEGER
    );

CREATE TABLE CYCLES (
    TYPE INTEGER,
    PER INTEGER
    );

CREATE TABLE TASK (
    TASK_TYPE         INTEGER,
    TASK_NUM          INTEGER NOT NULL,
    TASK_LEVEL        INTEGER NOT NULL,
    TASK_PARENTNUM    INTEGER NOT NULL,
    TASK_ROOTNUM      INTEGER
    TASK_ID           VARCHAR(20),
    TASK_NAME         VARCHAR(240),
    TASK_DUEDATE      INTEGER,
    TASK_FIXDATE      INTEGER,
    TASK_ESTDATE      INTEGER,
    TASK_STARTDATE    INTEGER,
```

```
    TASK_FINISHDATE   INTEGER,
    TASK_CYCLE        INTEGER,
    TASK_DURATION     INTEGER,
    TASK_LOGDURATION  INTEGER,
    TASK_PCTEFFORT    INTEGER,
    TASK_STATUS       INTEGER,
    TASK_KW           VARCHAR(20),
    TASK_KWPERF       VARCHAR(20),
    TASK_PRIORITY     INTEGER,
    TASK_ASSIGNBY     VARCHAR(20),
    TASK_CYCLETNUM    INTEGER,
    TASK_INSTPARENT   INTEGER,
    TASK_GROUP        VARCHAR(10),
    TASK_ORG          VARCHAR(20),
    TASK_PRIVATE      INTEGER,
    PCT_COMPLETE      INTEGER,
    SUBTASK_COUNT     INTEGER,
    STEP_COUNT        INTEGER,
    ATTACH_COUNT      INTEGER,
    DOIT_COUNT        INTEGER,
    TASK_TODO         INTEGER,
    TASK_LATESTART    INTEGER,
    STATUS_CHANGE      VARCHAR(20),
    TASK_DUEDATE_FMT   VARCHAR(255),
    TASK_DURATION_FMT  VARCHAR(255),
    TASK_PCTEFFORT_FMT VARCHAR(255),
    CYCLENUM          INTEGER
    );
CREATE UNIQUE INDEX TASK_NUM ON TASK(TASK_NUM);
CREATE INDEX TASK_PARENT_NUM ON TASK(TASK_PARENTNUM);
CREATE INDEX TASK_DUEDATE ON TASK(TASK_DUEDATE);
CREATE INDEX TASK_KW ON TASK(TASK_KW);
CREATE INDEX TASK_KWPERF ON TASK(TASK_KWPERF);
CREATE INDEX TASK_CYCLETNUM ON TASK(TASK_CYCLETNUM);
CREATE INDEX TASK_LATESTART ON TASK(TASK_LATESTART);
CREATE INDEX TASK_ROOTNUM ON TASK(TASK_ROOTNUM);

CREATE TABLE KWTASK (
    TASK_NUM    INTEGER NOT NULL,
    KW_ID       VARCHAR(20),
    TASK_STATUS INTEGER
    );
CREATE INDEX KWTASK_TASK_NUM ON KWTASK(TASK_NUM);
CREATE INDEX KWTASK_KW_ID ON KWTASK(KW_ID);

CREATE TABLE PERM (
    TASK_NUM     INTEGER NOT NULL,
    KW_ID        VARCHAR(20),
    DELETE_PERM  INTEGER,
    UPDATE_PERM  INTEGER
    );
CREATE INDEX PERM_TASK_NUM ON PERM(TASK_NUM);

CREATE TABLE STEP (
    STEP_NUM              INTEGER NOT NULL,
```

```
    STEP_TASK_NUM       INTEGER NOT NULL,
    STEP_ORDERNUM       INTEGER,
    STEP_NAME           VARCHAR(240),
    STEP_FINISHDATE     INTEGER,
    STEP_STATUS         INTEGER,
    ATTACH_COUNT        INTEGER,
    DOIT_COUNT          INTEGER
    );
CREATE UNIQUE INDEX STEP_NUM ON STEP(STEP_NUM);
CREATE INDEX STEP_TASK_NUM ON STEP(STEP_TASK_NUM);


CREATE TABLE STEP_STATUS (
    STEP_NUM            INTEGER NOT NULL,
    TASK_NUM            INTEGER NOT NULL,
    STATUS              INTEGER,
    FINISHDATE          INTEGER
    );
CREATE INDEX STEP_STATUS_STEP ON STEP_STATUS(STEP_NUM);
CREATE INDEX STEP_STATUS_TASK ON STEP_STATUS(TASK_NUM);
CREATE INDEX STEP_STATUS_STATUS ON STEP_STATUS(STATUS);


CREATE TABLE TASK_PRED (
    TASK_SUCC_ID    INTEGER NOT NULL,
    TASK_PRED_ID    INTEGER NOT NULL
    );
CREATE INDEX TASK_SUCC_ID ON TASK_PRED(TASK_SUCC_ID);
CREATE INDEX TASK_PRED_ID ON TASK_PRED(TASK_PRED_ID);
CREATE UNIQUE INDEX TASK_PREDIND ON TASK_PRED(TASK_SUCC_ID,
TASK_PRED_ID);



CREATE TABLE ATTACHMENT (
    ATTACHMENT_NUM      INTEGER NOT NULL,
    ATTACHMENT_TYPE     INTEGER,             /* AT_PUBLIC,
AT_PRIVATE, AT_REMOVABLE, AT_SENSITIVE */
    ATTACHMENT_TITLE    VARCHAR(80),
    ATTACHMENT_NAME     VARCHAR(130),        /* File name */
    ATTACHMENT_READONLY INTEGER,             /* NOT USED after
1.731 001 */
    ATTACHMENT_REMOVABLE INTEGER,            /* NOT USED after
1.731 001 */
    ATTACHMENT_VIEWER  VARCHAR(80),          /* NOT USED after
1.731 001 */
    ATTACHMENT_SYSFILE VARCHAR(20),          /* NOT USED */
    ATTACHMENT_FILE     LONG VARCHAR,        /* NOT USED */
    ATTACHED_BY         VARCHAR(20),
    ATTACHED_DATE       INTEGER,
    LASTEDIT_BY         VARCHAR(20),
```

```
    LASTEDIT_DATE        INTEGER,
    APPNAME              VARCHAR(80),
    PREV_VERSION         INTEGER,
    ORIG_VERSION         INTEGER,
    VERSION_NUM          INTEGER,
    REMOVABLE_TITLE      VARCHAR(80),
    REMOVABLE_DIR        VARCHAR(130),
    SENSITIVE            INTEGER
    );
CREATE UNIQUE INDEX ATTACHMENT_NUM ON
ATTACHMENT(ATTACHMENT_NUM);


CREATE TABLE ATTACHPERM (
    ATTACHMENT_NUM  INTEGER NOT NULL,
    KW_ID           VARCHAR(20),
    PERM            INTEGER
    );
CREATE INDEX ATTACHPERM_NUM ON ATTACHPERM(ATTACHMENT_NUM);


CREATE TABLE ITEM_ATTACHMENT (
    ATTACHMENT_NUM INTEGER NOT NULL,
    ITEM_NUM       INTEGER NOT NULL,
    ITEM_TYPE      INTEGER NOT NULL
    );
CREATE INDEX ITEM_ATTACH_NUM ON
ITEM_ATTACHMENT(ATTACHMENT_NUM);
CREATE INDEX ATTACH_ITEM_NUM ON ITEM_ATTACHMENT(ITEM_NUM);
CREATE INDEX ATTACH_ITEM_TYPE ON
ITEM_ATTACHMENT(ITEM_TYPE);
CREATE UNIQUE INDEX ITEM_ATTACH ON
ITEM_ATTACHMENT(ITEM_TYPE, ITEM_NUM, ATTACHMENT_NUM);


CREATE TABLE DOIT (
    DOIT_NUM       INTEGER NOT NULL,
    DOIT_COMMAND   VARCHAR(130),
    DOIT_TITLE     VARCHAR(80),
    DOIT_EXEC      INTEGER,
    DOIT_WORKDIR   VARCHAR(64),
    DOIT_PARAMS    VARCHAR(130)
    );
CREATE UNIQUE INDEX DOIT_NUM ON DOIT(DOIT_NUM);


CREATE TABLE ITEM_DOIT (
    DOIT_NUM    INTEGER NOT NULL,
    ITEM_NUM    INTEGER NOT NULL,
    ITEM_TYPE   INTEGER NOT NULL
    );
```

```
CREATE INDEX ITEM_DOIT_NUM ON ITEM_DOIT(DOIT_NUM);
CREATE INDEX DOIT_ITEM_NUM ON ITEM_DOIT(ITEM_NUM);
CREATE INDEX DOIT_ITEM_TYPE ON ITEM_DOIT(ITEM_TYPE);
CREATE UNIQUE INDEX ITEM_DOITIND ON ITEM_DOIT(ITEM_TYPE,
ITEM_NUM, DOIT_NUM);


CREATE TABLE KWS_SYSTEM (
    ATTACHMENT_DIR   VARCHAR(240),
    EXEFILE_NUM      INTEGER,
    VERSION          VARCHAR(20),
    DBID             VARCHAR(20)
    );


CREATE TABLE MESSAGE (
    MESSAGE_NUM   INTEGER,
    KW_ID         VARCHAR(20) NOT NULL,
    SENDER        VARCHAR(20),
    DATE_SENT     INTEGER,
    RECEIVED      INTEGER,
    SUBJECT       VARCHAR(80),
    MESSAGE       LONG VARCHAR,
    TIME_SENT     DATE
    );
CREATE UNIQUE INDEX MESSAGE_NUM ON MESSAGE(MESSAGE_NUM);
CREATE INDEX MESSAGE_KW_ID ON MESSAGE(KW_ID);


CREATE TABLE FIELD (
    KW_ID        VARCHAR(20) NOT NULL,
    FIELDSET_ID  INTEGER,
    FIELD_ID     INTEGER,
    FIELD_ORDER  INTEGER,
    WIDTH        INTEGER,
    WRAP         INTEGER
    );
CREATE INDEX FIELD_KW_ID ON FIELD(KW_ID);
CREATE INDEX FIELD_FIELDSET_ID ON FIELD(FIELDSET_ID);
CREATE UNIQUE INDEX FIELD_UNIQUE ON FIELD(KW_ID,
FIELDSET_ID, FIELD_ID);


CREATE TABLE DOCUMENT (
    NUM             INTEGER,
    ORDERNUM        INTEGER,
    DATA            LONG
    );
CREATE INDEX DOCUMENT_NUM ON DOCUMENT(NUM);
```

```
CREATE SEQUENCE DOCUMENT_SEQ;

CREATE TABLE WORKGROUP (
    ID        VARCHAR(20) NOT NULL,
    NAME      VARCHAR(240),
    ORG_ID    VARCHAR(20)
    );
CREATE UNIQUE INDEX WORKGROUP_ID ON WORKGROUP(ID);

CREATE TABLE WORKGROUP_ASSIGN (
    WORKGROUP_ID      VARCHAR(20) NOT NULL,
    KW_ID             VARCHAR(20) NOT NULL
    );
CREATE INDEX WORKGROUP_ASSIGN_GROUP ON
WORKGROUP_ASSIGN(WORKGROUP_ID);
CREATE INDEX WORKGROUP_ASSIGN_KW ON
WORKGROUP_ASSIGN(KW_ID);
CREATE UNIQUE INDEX WORKGROUP_ASSIGN_GROUPKW ON
WORKGROUP_ASSIGN(WORKGROUP_ID, KW_ID);

CREATE TABLE ORG (
    ID        VARCHAR(20) NOT NULL,
    NAME      VARCHAR(240)
    );
CREATE UNIQUE INDEX ORG_ID ON ORG(ID);

CREATE TABLE VIEWER (
    KW_ID          VARCHAR(20) NOT NULL,
    EXT            VARCHAR(10) NOT NULL,
    COMMAND        VARCHAR(80)
    );
CREATE INDEX VIEWER_KW_ID ON VIEWER(KW_ID);
CREATE UNIQUE INDEX VIEWER_KW_EXT ON VIEWER(KW_ID, EXT);

CREATE TABLE APPT (
    APPT_NUM       INTEGER NOT NULL,
    KW_ID          VARCHAR(20) NOT NULL,
    TITLE          VARCHAR(240),
    APPT_DATE      INTEGER,
    START_TIME     DATE,
    END_TIME       DATE,
    ATTACH_COUNT   INTEGER
    );
CREATE UNIQUE INDEX APPT_NUM ON APPT(APPT_NUM);
CREATE INDEX APPT_KW_ID ON APPT(KW_ID);
CREATE INDEX APPT_DATE ON APPT(APPT_DATE);
```

```
CREATE TABLE SEQUENCE_GEN (
    NEXTNUM    NUMBER
    );


CREATE TABLE PALETTE (
    NUM               INTEGER,
    NAME              VARCHAR(255),
    KWID              VARCHAR(20),
    X                 INTEGER,
    Y                 INTEGER,
    WIDTH             INTEGER,
    HEIGHT            INTEGER,
    DEFICON_NAME     VARCHAR(255),
    GROUPID           VARCHAR(20)
    );
CREATE UNIQUE INDEX PALETTE_NUM ON PALETTE(NUM);
CREATE UNIQUE INDEX PALETTE_KWID_NAME ON PALETTE(KWID,
NAME);


CREATE TABLE PALETTE_ENTRY (
    PALETTE_NUM      INTEGER,
    TASK_NUM         INTEGER,
    KWID             VARCHAR(20),
    TEMPLATE_TITLE VARCHAR(255),
    X                INTEGER,
    Y                INTEGER,
    ICON_NAME        VARCHAR(255)
    );
CREATE INDEX PALETTE_ENTRY_NUM ON
PALETTE_ENTRY(PALETTE_NUM);
```

Appendix B: KWS Organizational Entities

Process	A process is a top-down sequence of tasks that are part of an organization's business plan or mission. Processes are often flowcharted, and an analysis of processes can lead to business process reengineering.
Task	Each process is composed of tasks with due dates. All tasks within a particular process must be completed before that process can be considered finished. Each knowledge worker is usually responsible for completing one or more tasks. KWS is structured so each task can be divided into a series of smaller tasks. These "subordinate" tasks can, in turn, be decomposed into even smaller, or lower tasks. There is no limit to the depth this task structure can reach.
Milestone	A milestone is defined as a task with zero duration. Milestones are created as organization level significant dates. They are not assigned to a particular user, but to the organization as a whole.
Public Tasks	Tasks that "belong" to a process that is part of the organization are termed public tasks.
Private Tasks	In addition to public tasks, each knowledge worker may also insert their own set of tasks. These private tasks will appear along with the list of assigned tasks in the knowledge worker's ToDo List (explained below under DoIts). Private tasks can also be decomposed into a series of lower-level tasks. Top-level private tasks exist on the process level, and their subordinate task levels are numbered in the same fashion as explained above. Other than that they do not "belong" to processes, private tasks are identical to public tasks, contain the same data fields, and are denoted by a "P" in the Symbol field.
Parents, Siblings	In the interest of brevity and ease of comprehension, the convention of describing relationships within the KWS hierarchy with family terminology is used. Consider a scenario with a single process and multiple tasks under that process. The process is the parent of the tasks, which are the process's children. Likewise, each task may be a sibling to other tasks. Siblings are tasks that have the same parent. Two processes, therefore, are not siblings since they do not share a common parent task. Tasks belonging to different processes are not siblings.
Steps	KWS can assist a knowledge worker in completing a process on the ToDo List efficiently and on time by showing—at a glance—exactly what procedures need to be followed. KWS provides the mechanism of steps to guide a knowledge worker through the details of completing a task. Unlike tasks, steps have no Date Due fields. Instead, they are numbered in the order in which they appear in the window. Steps are different from processes and tasks; they outline the procedures necessary to complete a process. Any task in KWS can have a Steps List.
Attachments	KWS lets you store two types of job aides: Attachments and DoIts. Each milestone, process, task, or step stored in the KWS hierarchy can have one or more Attachments or DoIts associated with it. To perform a task, it may be necessary to follow a detailed sequence of instructions or view information related to that task. Attachments are files that can be linked to a particular item of work and viewed when necessary.
DoIts	DoIts are automated repetitive tasks often performed using a computer, such as a multistep calculation or report generation. Adding a DoIt allows the user to execute a program or series of programs directly from KWS. The user can execute DOS batch or executable files, as well as any Windows applications.

Abbreviations and Acronyms

BPI	Business Process Improvement
BPR	Business Process Reengineering
CEAP	Corps of Engineers Application Program
CPM	Critical Path Method
CRDA	Cooperative Research and Development Agreement
DDA	Dependent-Duration-Activity
GTM	Graphical Task Manager
HQUSACE	Headquarters, U.S. Army Corps of Engineers
IDEF	Integrated Definition Language
KWS	Knowledge Worker System
PERT	Program Evaluation and Review Technique
SOW	Scope of Work
TQM	Total Quality Management
USACERL	U.S. Army Construction Engineering Research Laboratories
WBS	Work Breakdown Structure

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CECC-R
ATTN: CECW
ATTN: CERM
ATTN: CEMP-MC (12)
ATTN: CERD-ZA

CECPW 22310-3862
ATTN: CECPW-E
ATTN: CECPW-FTT
ATTN: CECPW-ZC

OASD 22202
ATTN: C3I (6)

US Army Engr District
ATTN: Wilmington 28402

US Army Engr Division
ATTN: Library (11)

Defense Distribution Region East
ATTN: DDRE-WI 17070

US Army Materiel Command (AMC)
Redstone Arsenal 35809
ATTN: DESMI-KLF
Rock Island Arsenal 61299
ATTN: SMCRI-EH
ATTN: SMCRI-TL

FORSCOM
Fort Hood 76544
ATTN: AFZF-DE-AES Engr

TRADOC
Fort Eustis 23604
ATTN: DPW
Fort Sill 73503
ATTN: ATZR-E

CEWES 39180
ATTN: Library

CECRL 03755
ATTN: Library

Engr Societies Library
ATTN: Acquisitions 10017

Defense Logistics Agency
ATTN: DLA-WI 22304

American Public Works Assoc. 64104-1806

US Gov't Printing Office 20401
ATTN: Rec Sec/Deposit Sec (2)

Nat'l Institute of Standards & Tech
ATTN: Library 20899

Defense Tech Info Center 22304
ATTN: DTIC-O (2)

58
+1
2/96

DEPARTMENT OF THE ARMY
CONSTRUCTION ENGINEERING RESEARCH LABORATORIES
CORPS OF ENGINEERS
PO BOX 9005
CHAMPAIGN, ILLINOIS 61826-9005

OFFICIAL BUSINESS

BULK RATE
US POSTAGE
PAID
CHAMPAIGN IL
PERMIT NO. 871